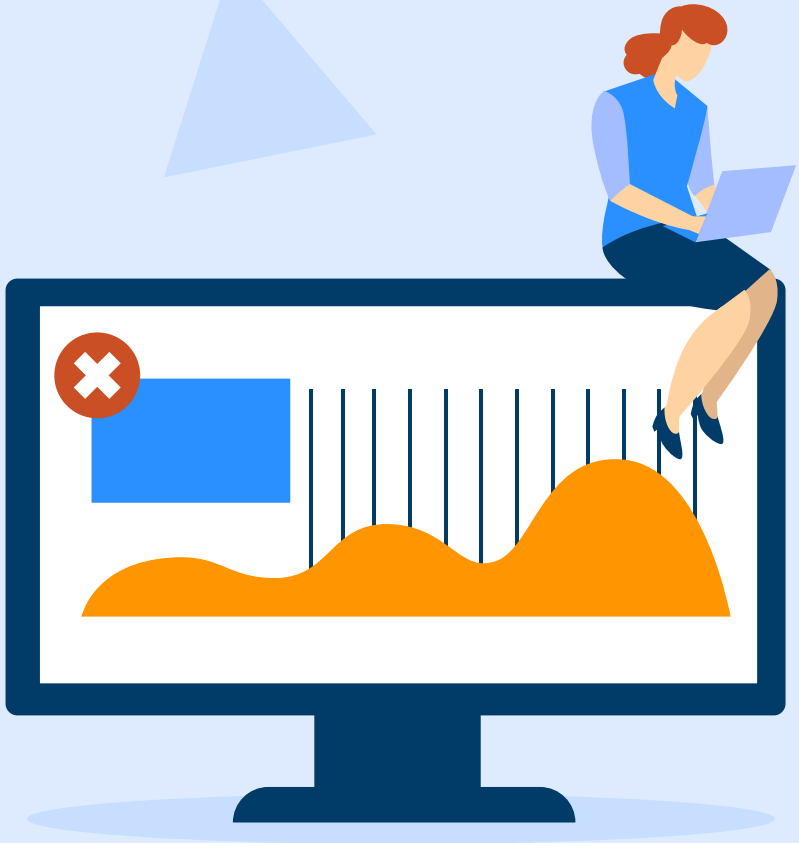
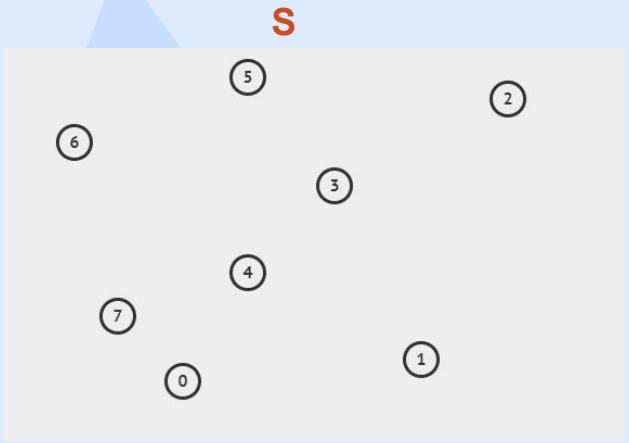
Geometría Computacional

Ailyn Rebollar Pérez

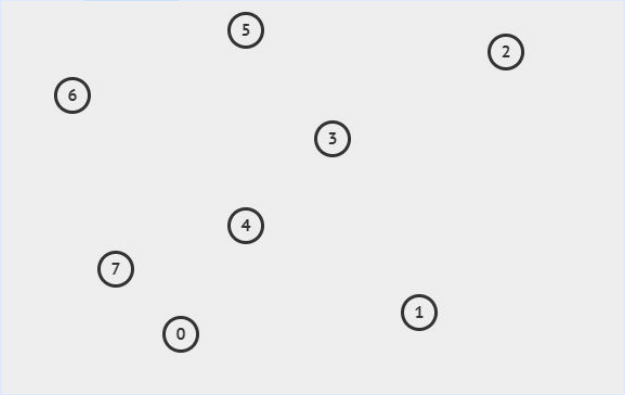


El **cierre convexo** de un conjunto de puntos S es el conjunto convexo más pequeño que contiene a S.

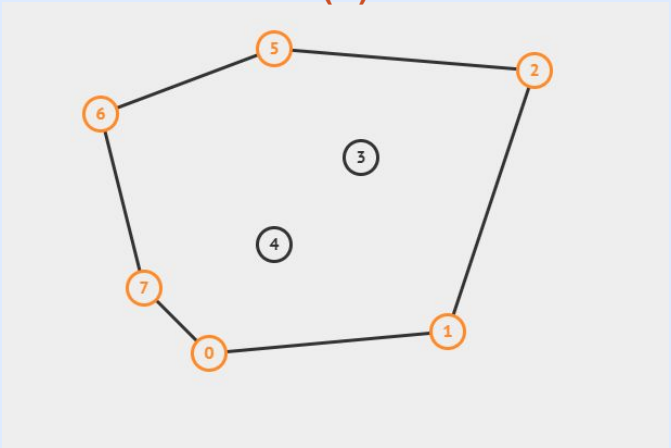


El **cierre convexo** de un conjunto de puntos S es el conjunto convexo más pequeño que contiene a S.

S



CH(S)



El **cierre convexo** de un conjunto de puntos S es el conjunto convexo más pequeño que contiene a S.

Graham Scan

Graham Scan

01

Encontrar al más chaparro

Graham Scan

01

Encontrar al más chaparro

02

Ordenar puntos

Graham Scan

01

Encontrar al más chaparro

02

Ordenar puntos

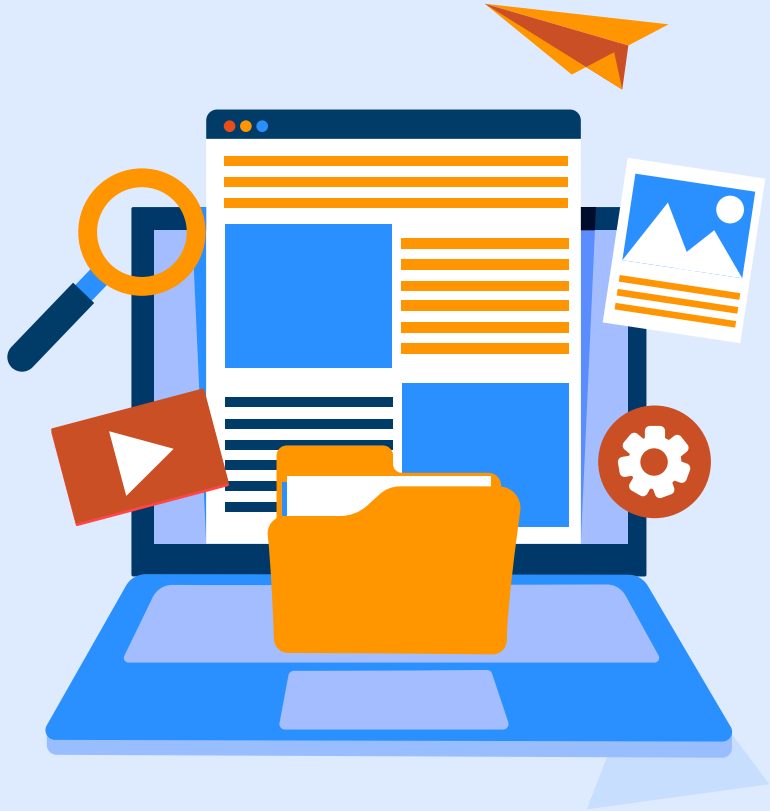
03

Scanner

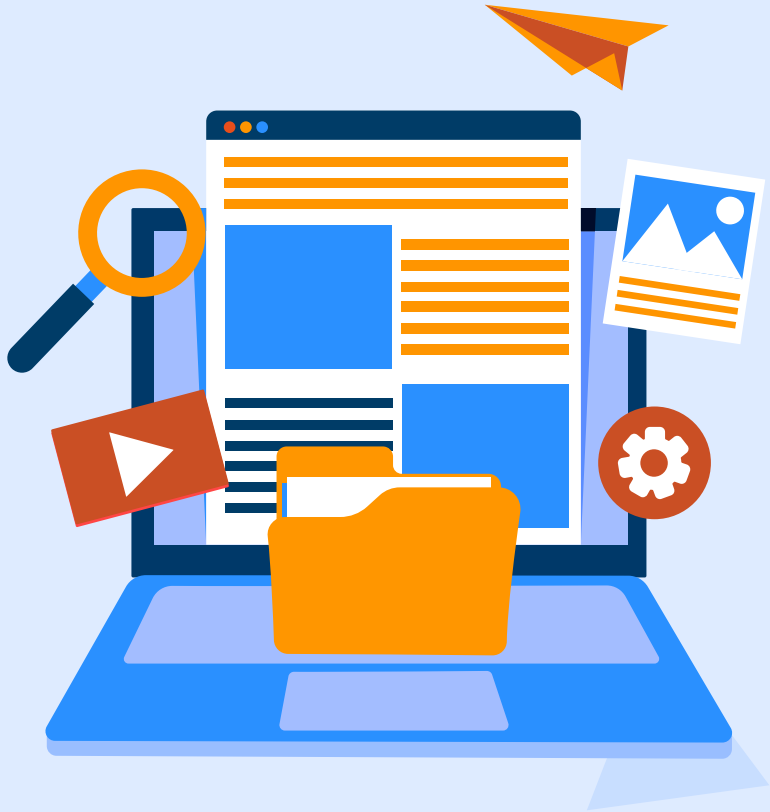
Recordemos el algoritmo

Hagamoslo mediante la liga:

https://visualgo.net/en/convexhull

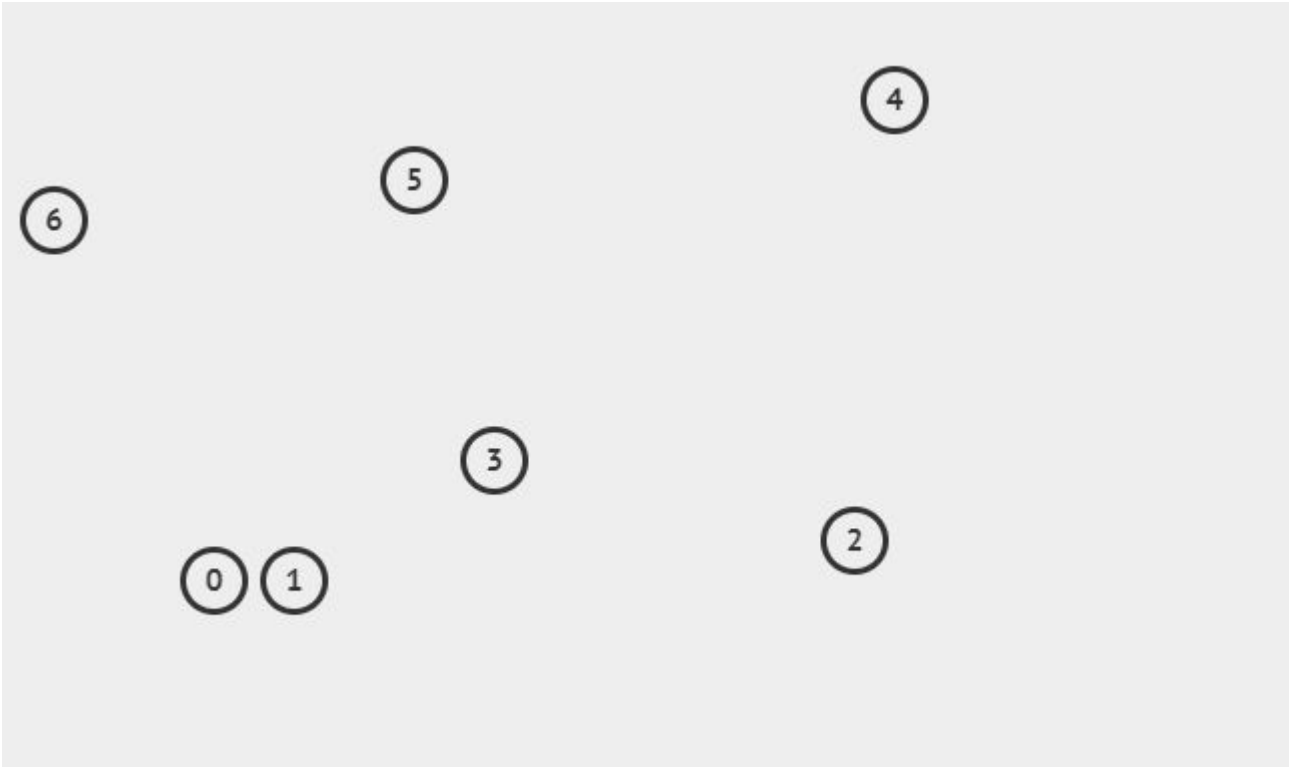


¿Es igual de fácil implementar el algoritmo?



01

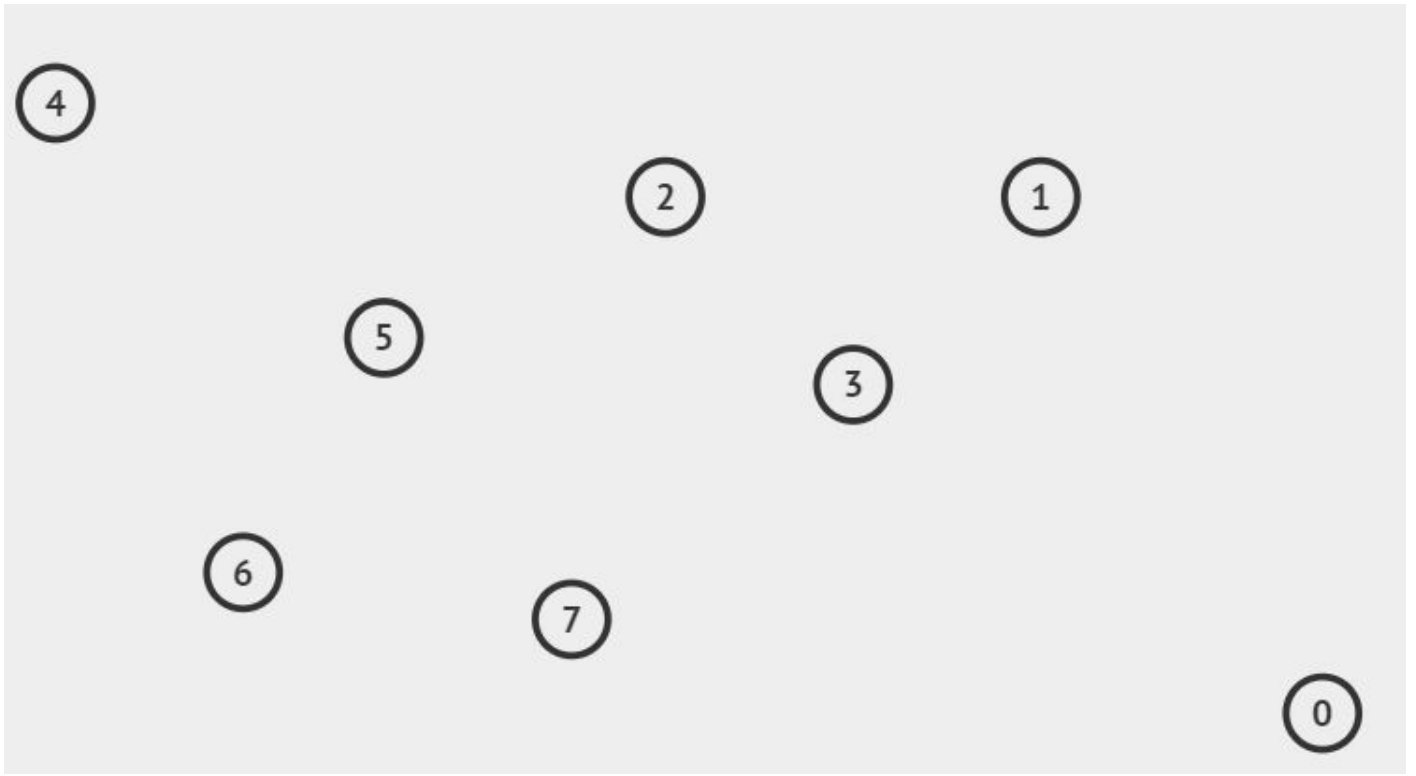
Encontrar al más chaparro



¿Qué sucede si tenemos dos puntos con la misma coordenada y?

02

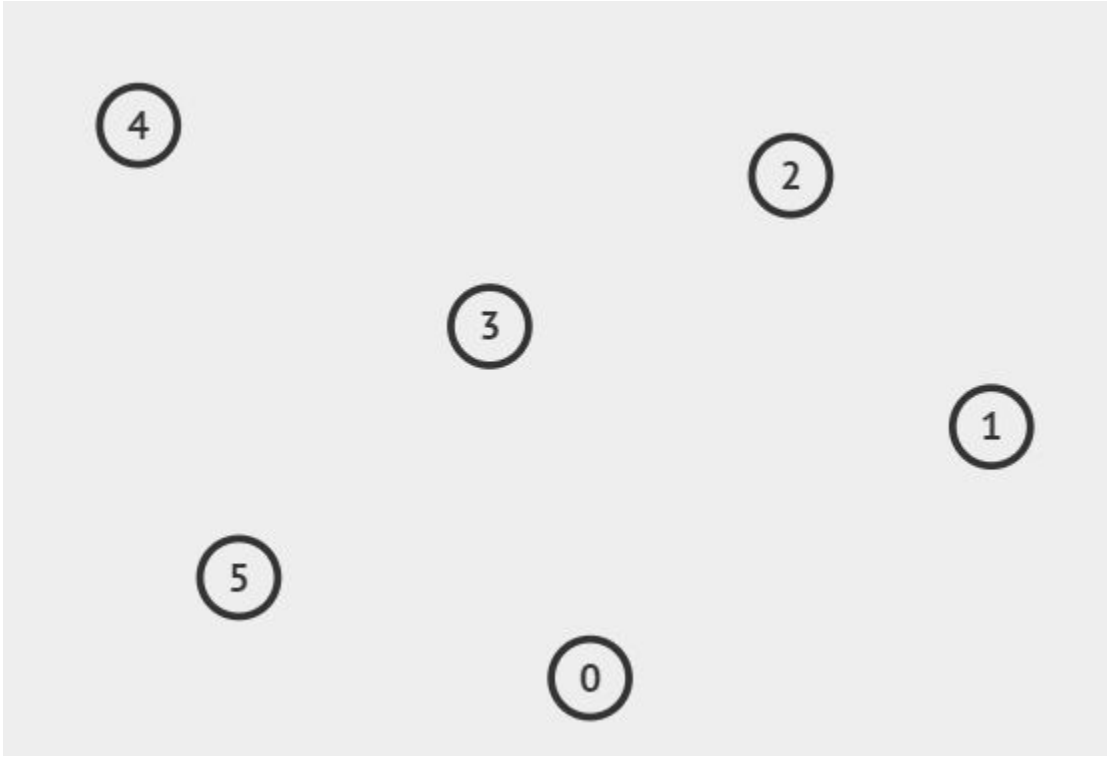
Ordenar puntos



¿Qué sucede si tenemos dos puntos colineales?

03

Scanner



¿Se tiene que realizar el scan en una dirección particular?

Pseudo-código

F. P. Preparata and M. I. Shamos. Computational Geometry: An Introduction. Springer-Verlag, New York, NY, 1985.

procedure GRAHAMHULL(S)

1. Find an internal point q.
2. Using q as the origin of coordinates, sort the points of S lexicographically by polar angle and distance from q and arrange them into a circular doubly-linked list with pointers NEXT and PRED, with START pointing to the initial vertex. The boolean parameter f, when true, indicates that vertex START has been reached through forward scan, not backtracking.
3. (Scan)

```
begin v := START; w := PRED[v]; f := false;
      while (NEXT[v] ≠ START or f = false) do
        begin if NEXT[v] = w then f := true;
              if (the three points v, NEXT[v], NEXT [NEXT[v]] form a left
                turn) then v := NEXT[v]
              else begin DELETE NEXT[v];
                     v := PRED[v]
                   end
        end
```

Pseudo-código

Joseph O'Rourke. Computational Geometry in C (2nd ed.). Cambridge University Press, 1998.

Algorithm: GRAHAM SCAN, VERSION A
Find interior point x; label it p_0.
Sort all other points angularly about x; label $p_1, \ldots, p_{n-1}$.
Stack $S = (p_2, p_1) = (p_t, p_{t-1})$; t indexes top.
$i \leftarrow 3$
while $i < n$ do
    if p_i is left of (p_{t-1}, p_t)
        then Push (p_i, S) and set $i \leftarrow i + 1$.
        else Pop(S).

Algorithm 3.5 Graham Scan, Version A.

Pseudo-código

Joseph O'Rourke. Computational Geometry in C (2nd ed.). Cambridge University Press, 1998.

Algorithm: GRAHAM SCAN, VERSION B
Find rightmost lowest point; label it p_0.
Sort all other points angularly about p_0.
    In case of tie, delete the point closer to p_0
    (or all but one copy for multiple points).
Stack $S = (p_1, p_0) = (p_t, p_{t-1})$; t indexes top.
$i = 2$
while $i < n$ do
    if p_i is strictly left of $p_{t-1}p_t$
        then Push(p_i, S) and set $i \leftarrow i + 1$
        else Pop(S).

Algorithm 3.6 Graham Scan, Version B.